(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 135 753**
A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **84109396.6**

(22) Date of filing: **08.08.84**

(51) Int. Cl.⁴: **G 06 F 9/44**, G 06 F 9/26

(30) Priority: **29.08.83 US 527053**

(43) Date of publication of application: **03.04.85 Bulletin 85/14**

(84) Designated Contracting States: **DE FR GB**

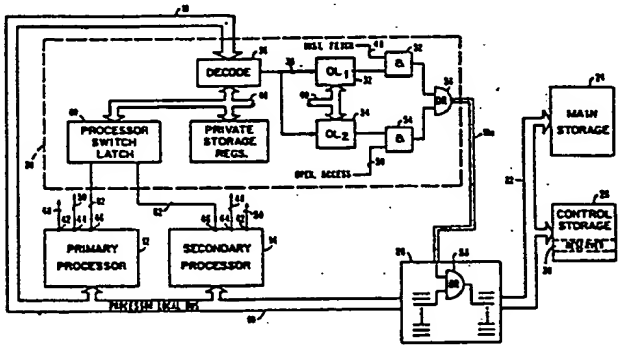
(71) Applicant: **International Business Machines Corporation, Old Orchard Road, Armonk, N.Y. 10504 (US)**

(72) Inventor: **Buonomo, Joseph Patrick, 250 Boswell Hill Road, Endicott New York 13760 (US)**
Inventor: **Houghtalen, Steven Ray, 945 N. McKinley Avenue, Endicott New York 13760 (US)**
Inventor: **Losinger, Raymond Ellison, 204 Ridgefield Road, Endicott New York 13760 (US)**
Inventor: **Valashinas, James William, 338 Boswell Hill Road, Endicott New York 13760 (US)**

(74) Representative: **Teufel, Fritz, Dipl.-Phys. et al, IBM Deutschland GmbH. Europäische Patentdienste Postfach 265, D-8000 München 22 (DE)**

(54) Storage selection override apparatus for a multi-microprocessor implemented data processing system.

(57) The performance of a multimicroprocessor implemented data processing system that emulates a mainframe is enhanced by providing a pair of override latches (32, 34) that serve to steer accesses between main and control storage for instruction fetch and operand acquisition in a manner that minimizes the complexity and size of microprocessor interface microcoding. This is achieved by connecting the instruction and operand override latches between a primary microprocessor (12), a secondary microprocessor (14), off-chip control storage (26) belonging to the secondary microprocessor, particularly memory mapped private storage therein, and main storage (24). The override latches are made responsive, via microcode provided for that purpose, to the type and cause of each memory access. The override latches are set or reset by a memory mapped write to a predefined address in the secondary control store after being enabled by control lines (48, 50) responsive to the particular microprocessor action being taken. When set, the instruction override latch directs all expected primary processor main storage instruction fetches to control store. When set, the operand override latch directs all expected primary processor main storage operand accesses to control store. As appropriate for instruction execution, either one or both of the the primary or secondary microprocessors can thereby be transparently latched to main or control storage.



EP 0 135 753 A2

STORAGE SELECTION OVERRIDE APPARATUS FOR A MULTIMICROPROCESSOR IMPLEMENTED DATA PROCESSING SYSTEM

This invention relates to a data processing system in accordance with the preamble of claim 1; it is concerned with storage utilization and selection apparatus for a multi-microprocessor implemented data processing system that emulates a main frame system. More particularly, this invention is directed to improving the performance of such a system insofar as selection between control and main storage is concerned.

The emulation of "mainframe" data processing systems through the use of microprocessors has become a reality. A typical main frame data processing system would be any one of the System/370 (S/370) models available from International Business Machines Corporation. The PC/XT370, a "desktop" System/370, also available from International Business Machines Corporation, is one example of such a microprocessor implemented main frame. This particular desktop system is a hardware/software package that allows one to run System/370 application programs in a single user environment, to run as a terminal attached to a main frame host or to run in a stand-alone mode, as required by the particular application. There are, of course, similar systems available from other manufacturers, all of which systems incorporate many of the same functions as the PC/XT370 although the manner and means of implementation does differ, in varying degrees, from system to system.

Due to revolutionary advances in chip densities and packaging, which have been accompanied by significant reductions in costs, many main frame features can now be implemented directly in a desktop system, while

other features require some hardware and/or software assistance in order to make them available. The introduction and use of more powerful microprocessors such as, for example, the 8086 and 8088 from Intel Corporation and the 68000 from Motorola Corporation, added further to the list of functions it would be possible to implement in a desktop mainframe. This new breed of microprocessors is fully capable of running a large, enriched instruction set, such as that of System/370, although several of these microprocessors, working in concert with the aid of additional hardware and/or software support, would be required to effect instruction execution in an acceptable time period. It will also be appreciated that presently available microprocessors, while remarkable for the functions they do offer, are not capable of providing all mainframe capability without system compromise.

Thus, as in all data processing system designs, various trade-offs are made in order to optimize the price and performance of these microprocessor implemented desktop mainframes. One particular trade-off problem is posed by the need or desire to utilize certain mainframe functions and features that would be particularly difficult to provide in a microprocessor implemented desktop mainframe. Another type of trade-off problem is posed by the requirement that all architectural constraints of the emulated mainframe be adhered to so that user programs can be run without concern. One specific implementation problem of concern, due in part to such trade-offs being made, is that of accommodating the power and versatility of the relatively large control storage of a mainframe central processing unit (CPU) in the relatively small control storage of a microprocessing unit (MPU). In the System/370 world, for example,

control store on the Model 138 is about 50 times larger than that available on the typical MPU chip. This disparity in control store capacity puts a premium on the amount and type of microcode that is placed into the MPU's control store.

This significant difference in the capacity of mainframe and MPU control storage capacity means that the performance of the MPU implemented mainframe system will be adversely affected each time an instruction or operation needs to be handled by reference outside of an MPU's on-chip control store. This control store capacity related performance problem is further exacerbated in a multi-MPU implemented system whenever the instruction fetching MPU passes control to another MPU for instruction execution or to obtain architectural information that is resident in control store or in the fetching MPU's registers. It should be remembered that in System/370 apparatus, and in many other mainframe systems as well, main storage is architecturally defined as belonging exclusively to the user. It cannot be contaminated by the system control program itself or by an interprocessor interface and is therefore unavailable for control or scratchpad functions. Further, due to the address bus limitations inherent in current MPUs, there are not sufficient bits to define independent equivalents of the mainframe's virtual and control storage modules, it is not practical to assign one or more of the address lines for the exclusive purpose of steering accesses between main and control storage. Of course, a mainframe system would not have to deal with these problems since it has sufficient addressing capability for its needs and possesses a control store module that is large enough to accommodate all of its microcode and to provide the scratchpad area it requires.

In the multi-MPU implemented mainframe data processing system, the lack of large control storage space can be overcome using specially written microcode to handle the interface between the microprocessors, by hardware assists or through some combination thereof. Further, off-chip control store can be readily provided. This approach would require the use of special and additional microcode to support the enhanced interprocessor interface or hardware assists. However, this additional microcode would require on-chip control store residence which would put less instruction responsive microcode in the fetching microprocessor's control storage, a result that would adversely affect overall system performance. Thus, while it would be possible to accommodate the MPU's smaller control store and compensate for the problems it presents, the performance and/or cost penalties associated with straightforward solutions to that problem are too great to accept.

It is therefore the object of the present invention to provide a data processing system of the aforementioned kind that will permit to optimally emulate a mainframe system having relatively large control storage, although its own control storage is only of limited capacity.

This object is achieved by the invention as characterized in claim 1; embodiments are characterized in the dependent claims.

The microprocessor access to main or control storage may be rerouted by override latches to the other storage type to minimize processor interface microcode and enhance system performance. Controlling and steering means are provided for storage accesses to optimize system performance transparent to the microprocessor seeking

said storage access. Controlling and steering storage accesses in a multi-microprocessor implemented mainframe data processing system provided here does not violate the architectural protective rules concerning mainframe storage accesses.

The invention proposes in a multi-microprocessor implemented mainframe emulated data processing system to connect instruction and operand override latches between a primary microprocessor, a secondary microprocessor, off-chip control storage belonging to the secondary microprocessor and main storage. The override latches are made responsive, via processor interface microcode provided for that purpose and processor control logic means, to the type and cause of each memory access. The override latches are set and reset by a memory mapped write to a predefined address in the secondary control store. When set, the instruction override latch directs all expected main storage instruction fetches by the primary microprocessor to a preselected area in the secondary processor's control store. When set, the operand override latch directs all expected main storage operand access to a preselected area of the secondary processor's control store. As appropriate for execution performance, either one or both of the the primary or secondary microprocessors can thereby be transparently latched to access either main or control storage although such access would not ordinarily be expected.

The invention will be described further, by way of a preferred example thereof, with reference to the accompanying drawings wherein like reference numerals have been used in the several views to depict like elements, in which:

Figure 1 schematically illustrates a simplified block diagram of a multimicroprocessor implemented mainframe data processing system which includes control and main memory storage; and

Figure 2 schematically depicts, in greater detail and in accordance with the present invention, latching means for steering memory accesses in the Figure 1 apparatus between main and control storage and some associated logic means.

The present invention is to be explained in the context of a mainframe desktop system that has been implemented with at least two microprocessors. More particularly, this resultant system has been adapted to emulate a System/370 mainframe. For those requiring further information on the instruction set of this mainframe and details of System/370 functions, reference should be made to the IBM System/370 Principles Of Operation (Manual No. GA22-7000), which is available from the IBM Corporation and which is, to the extent necessary, incorporated herein by reference. In addition, those requiring further information on the details of the desktop mainframe referred to herein should refer to Technical Reference Manual For The IBM PC XT/370 (Manual No. XXxx-xxxx).

It will be understood by those having skill in this art that mainframe implementation can be achieved by use of only a single microprocessor. Alternatively, a plurality of microprocessors, equal to or different than the number used herein, could be employed to emulate a mainframe system. Further divergence in system configuration is possible as a result of variations in instruction

set partitioning schemes and the manner in which the subsets are then emulated. Examples of this multiple microprocessor implementation approach are more completely described in the European Patent application 92610 of December 20, 1982. In this patent application, a System/370 instruction set is partitioned in accordance with several criteria and the subsets thereof are each implemented on one or more of a plurality of microprocessors, but not all necessarily in the same manner.

An illustrative desktop mainframe data processing system 10 is shown in Figure 1. As depicted in the simplified system block diagram thereof, a primary processing unit 12, and its associated secondary microprocessors 14 and 16 are connected to a local processor bus 18. Local bus 18 is connected, in turn, by bus-to-bus adapter 20 to the system bus 22. Main storage 24 and the secondary control storage 26 are both connected to the system bus 22. The primary processor 12 and secondary processors 14 and 16 are also responsively connected to processor control logic means 28 which incorporates processor control and interface logic and some private storage therefor. Certain aspects of the control logic means 28 shall be discussed hereinafter in greater detail.

In the particular embodiment described herein, primary processor 12 is assigned the responsibility for performing all instruction fetches and operand address calculations for all of the processors used in the system. It also performs execution of all fixed point instructions, contains and maintains the general purpose registers, instruction length codes, condition codes and instruction addresses, recognizes system interrupts and provides indications to the system that a main storage instruction fetch or operand access is required.

In addition, primary processor 12 is also able to provide an indication to the system that a change in processor control is needed.

Secondary processor 14 performs execution of all system control instructions, contains and maintains all of the control registers, when necessary performs the service processor function and provides indications to the system of main storage operand access and private storage microcode access. In addition, secondary microprocessor 14 is additionally able to provide the system with an indication that a change in processor control is needed.

Secondary microprocessor 16 performs execution of all floating point instructions, containing and maintaining all of the floating point registers. It also provides the system with an indication of main storage operand access and of a need to alter microprocessor control. Alternatively, these floating point functions can be provided by a a peripheral unit rather than by a microprocessor.

The mainframe instruction set is thus allocated for execution among the several processors. Primary processor 12 is provided with limited on-chip control store that can be utilized to store mainframe instruction responsive microcode and/or microprocessor interface and control microcode. It will be recognized, given the fixed quantity of on-chip control store available, that the instruction responsive microcode and the interface microcode reside in control store at the cost of the other. A greater amount of one type of microcode in on-chip control store residence means that a lesser amount of the other type can be accommodated therein. If a more functional microprocessor interface is

desired, with an attendant cost in supporting microcode, there will be less room in control store for instruction responsive microcode. From a performance standpoint, it is best to keep the interface simple and leave as much control store as possible for instruction code. The present invention facilitates and makes this possible. In this embodiment, for example, it has been decided to place microcode for the most frequently used mainframe instructions in the control store of microprocessor 12 and to use a relatively simple intermicroprocessor interface that requires minimal microcode.

A main storage module 24 is attached to system bus 22 and used as needed by the processors 12, 14 and 16. It is assumed that the system bus 22 and the microprocessors 12, 14 and 16 all include 24 bits of addressing to accommodate the addressing structure of the mainframe to be implemented. It may be necessary to slightly modify currently available microprocessors to achieve this addressing capability. The secondary processor 14 uses off-chip control storage module 26, as may be necessary, for its own microcode and scratchpad functions. While secondary processor 16 has no need to use off-chip control store 26, in this embodiment, it could access that module, as might be necessary, to satisfy its microcode and scratchpad needs. Processors 12, 14 and 16 and processor control logic means 28 are interconnected together by and pass information to each other on the processor local bus 18.

Because all of the available address bits or lines in a microprocessor implemented mainframe will be needed to define and emulate the mainframe's virtual storage, it would not be effective to divide all possible storage defined by the available address bits between virtual

main storage and control storage. Since all of the available address lines are needed to define virtual storage, prior to calculation of the real address involved, there is no direct manner of using those same address lines to also identify unique control storage addresses.

Although shown as two separate modules, and they are from a logical standpoint, main storage and control storage are a physically contiguous block of random access memory (RAM), with an exception to be discussed below. The dividing line between storage modules, as described herein, is the dividing line between real main storage and control storage. In this illustrative embodiment, the main storage module 24 runs from address 00000 to address 7FFFF (hexidecimal - hereinafter hex). The control storage module 26 runs from address 80000 to address 9FFFF (hex). It should be noted that bit 19 of any memory address placed on system bus 22, where bit 23 of the address is the most significant bit thereof, determines which memory module is accessed. If bit 19 of an address is set, then the address sought is 80000 (hex) or higher and will therefore be found in the off-chip control store module 26. If bit 19 is off, then the address sought is 7FFFF (hex) or lower and will therefore be found in the main memory module 26. The addresses used herein have been selected to simplify and facilitate this description. Those having skill in this art will recognize that the address limits for each memory module are a design choice and that the manipulation of more than than one address bit, to steer between main and control storage, may be necessary.

Private store 30, referred to previously, is logically a portion of off-chip control storage 26, but is physic-

ally located in the processor control logic means 28 and mapped into a reserved segment 26a of control store 26. The reserved segment 26a of control store 26 is typically about 256 bytes long, although it can be greater. As shall be discussed in connection with Figure 2, the processor control logic means 28 is connected to bus-to-bus interface 20 via bus feeder 18a. Also physically located in the processor control logic means 28 are a pair of override latches 32 and 34 that serve to steer memory accesses from processors 12 and 14 to either the main memory storage module 24 or to the off-chip control storage module 26 as shall be hereinafter explained in greater detail.

The function of latches 32 and 34 is shown more explicitly in Figure 2 wherein several system elements not directly pertinent to the present invention have been omitted to enable a more explicit description thereof. The latches are identical standard, off-the-shelf parts available from several vendors under different part numbers. For example, the MC74LS175N latches manufactured by Motorola Inc.'s Semiconductor Products Division would be suitable for use as the latches 32 and 34 as would the SN74LS175J type available from Texas Instruments Inc. There are four latches on each chip, only two of which are used for override purposes. Latch 32 serves as an instruction override latch and is set and reset by a memory mapped write to private store 30. When set, it causes all main storage instruction fetches by primary processor 12 to be directed to a designated area of control store 26 instead of main storage 24. Latch 34 serves as an operand override latch and is also set and reset by a memory mapped write to private store 30. When set, it causes all main storage operand accesses by primary processor 12 or secondary processor

14 to be respectively directed to a designated area of main storage 24 or control store 26. Including override or storage selection latches permits the implementation of many useful functions without requiring modification of other portions of the system, particularly to the on-chip microcode for microprocessor 12. It would also be possible to use override latches to alter all memory accesses by any system microprocessor if that capability were appropriate to the needs of the particular system involved.

The following example of how the latches work and the system context of their operation will best explain their role and usefulness. The interface between the primary and secondary microprocessors 12 and 14 is defined by appropriate microcode stored in control store module 26. On power-up, this microcode is accessed in typical fashion and the secondary microprocessor 14 is thereby given control. As part of this initialization process, secondary microprocessor 14 places startup information for primary microprocessor 12 at predefined control storage 26 addresses and resets override latches 32 and 34, which forces them to a logical 0. Control is then passed to the primary microprocessor 12 by the processor interface code, once the housekeeping chores are accomplished.

The latches are set or reset only by the secondary microprocessor 14 although that function need not be so restricted. This action occurs whenever a "write" operation of latch data is attempted to a preassigned address in private store 30 by processor 14. When that occurs, the decode circuit means 36 decodes the address on processor bus 18 and, in response thereto, raises its latch enable line 38 to latches 32 and 34. When enable line 38 is raised, one bit of the data latch information available

on data bus input 40 to each latch, and this is a different bit for each latch although the value of the bits may be identical, is thereby gated into the respective latches 32 and 34. If the gated data bit is a logical 0, the latch is said to be reset. If the data bit is a logical 1, the latch is said to be set.

Each of the microprocessors 12 and 14 are provided with a number of input/output (I/O) and control pins, as specified by the manufacturers. Among these are an instruction fetch pin 42, an operand access pin 44 and a bus grant acknowledge (BGACK) pin 46. Signals present on these pins, and the control lines connected thereto, indicate respectively that the particular microprocessor involved is performing an instruction fetch, attempting an operand access or going to an idle state because another device has been granted bus access. Thus, when instruction fetch line 48 is set to a logical high, the microprocessor involved is indicating that it is going to perform an instruction fetch. Likewise, when operand access line 50 goes high, it means that a microprocessor is about to access memory to retrieve an operand. For a purpose to be discussed later, receipt of a signal setting the BGACK pin 46 to a logical 1 or high, indicates that another device has been granted access to bus 18 and forces the receiving processor to an idle state.

Since the instruction fetch and operand access operations are exclusive and do not occur at the same time, only one of the control lines 48 and 50 can be set at the same time. When the latch enable line 38 goes high, data is read into latches 32 and 34. The output of latches 32 and 34 are fed to AND gates 52 and 54 respectively. The other input to AND gate 52 is the instruction fetch control line 48. The other input to AND gate 54 is the

operand access control line 50. The output of AND gates 52 and 54 is forwarded, in turn, to OR gate 56. Thus, if either instruction override latch 32 or operand override latch 34 has a logical 1 output and its associated control line is similarly set, then the output of OR gate 56 will be set to a logical 1. The output of OR gate 56 is connected to the input of address OR gate 58. The other input to OR gate 58 is bit 19 of the address being sought in memory. If the output of either AND gate 52 or 54 is set, bit 19 of the memory address is set, thereby forcing a read from or write into the control storage module 26. This is done transparently to the processor in control and works a non-expected access of control storage module 26.

As previously noted, the primary microprocessor 12 is adapted to perform all system instruction fetches. It is also provided with its own on-chip microcode that enables it to directly execute a number of the most frequently used mainframe instructions. As long as it can do so, without assistance from or intervention by secondary processor 14, primary microprocessor 12 will continue to execute its "own" instructions. However, when an interrupt occurs or when primary processor 12 encounters an op code it is not microprogrammed to handle, it initiates a secondary processor call routine that ultimately turns on secondary processor 14.

Transfer of control between primary microprocessor 12 and secondary microprocessor 14 is handled in accordance with microcode provided for that purpose. In the case of primary microprocessor 12 that microcode is resident on-chip. The control transfer routine for secondary microprocessor 14 is found in control storage 26. It will first be assumed that primary microprocessor 12

has control and encounters an op code it cannot handle or that it receives an interrupt. In either event, primary microprocessor 12 responds by storing key information in private store 30. This key information includes the contents of the transferring processor's program counter and program status register and a calling code which indicates the reason why processor control transfer was initiated. If necessary for use by the receiving processor, the transferring processor will also store operands, the current instruction, bus error information or any other required parameters. After this critical status information is stored, the processor control transfer routine then concludes by attempting a write to a predefined address, that of the processor switch latch 60, in private store 30 followed by a read of another predefined private store address. Upon receipt of the write address, decoder 36 determines that latch 60 is being addressed and enables that device causing the BGACK line 62 of the processor relinquishing control to be set and that of the processor getting control to be reset. The following read command is thus never executed, but remains in the input pipeline of the processor that has now relinquished control.

When this control sequence is again invoked, the BGACK pin 46 of the idle processor is reset, causing it to become active and regain control. The first thing that the processor does is complete the read operation it had previously been requested to perform. Depending upon the particular microprocessor regaining control and on the nature of the incident that previously triggered control transfer, the read address may begin a reload of the information stored by the primary microprocessor 14 or it could effect a read of the calling code and initiate responsive action thereto.

The period for which control is relinquished by primary processor 12 may be for only one instruction. This is accomplished by setting the trace bit of the control transferring processor's program status register (PSR) before its contents are placed in private store 30 and then having the control acquiring processor read that information into its own PSR. Alternatively, it is possible to transfer control from the primary processor 12 to the secondary processor 14 after only one or any other number of predetermined instructions have been executed by utilizing an invalid op code that causes an interrupt when encountered. The interrupt, of course, invokes secondary processor control as per system definition.

The entire process of control transfer and utilization of control storage 26 by the primary microprocessor 12, as facilitated by the override latches 32 and 34, would work as follows in the case of a requirement to execute a System/370 execute instruction received by primary processor 12. The System/370 execute instruction causes a target instruction found at the address pointed to by its second operand to be modified by the contents of the general register specified in the register field of the execute instruction, after which the target instruction is executed. It will be appreciated by those having skill in the appertaining art that under System/370 architectural constraints, main memory cannot be used or corrupted by system needs. Thus, the need to store intermediate results or a copy of an instruction, the target instruction in this instance, must be satisfied by control storage utilization only.

The primary processor 12 is in control of the system and fetches its next instruction, which happens to be

the execute instruction. This is one of the instructions it is not microprogrammed to handle. It computes the address where the target instruction can be found in main storage 24 and then, knowing from its microcode that it cannot proceed further, primary processor 12 writes that address, the appropriate calling code explaining the reason for control transfer and other pertinent register information into private store 30, before passing control to secondary processor 14. The secondary processor 14 then examines the calling code and, based on its microcoded response to an execute instruction, writes a copy of the subject instruction into control store 26 using the information left in private store 30 by primary processor 12. The second byte of the copied instruction is then ORed with low order byte of the GPR value passed from primary processor 12 and stuffed back into the second byte of the control store copy of the subject or target instruction.

In this example, it is further assumed that the subject instruction is a "move character" instruction that is the target of the original "execute" instruction. The move character instruction is one that has been defined as being executable by primary processor 12. When that fact is recognized by the secondary processor 14 from the op code of the target instruction, it saves the original contents of the program counter passed to it by primary processor 12 and replaces the address of the next instruction primary processor 12 would have fetched, but for control transfer, with an address for the now modified move character instruction. Although that modified instruction is resident in control store 26, address bit 19 of that address is reset to 0. Thus, primary processor 12 will be looking for the move char-

acter instruction in main storage 24, rather than in control storage module 26. Note that the present invention allows primary processor 12 to execute one of its directly executable instructions as a target of a System/370 execute instruction without requiring any special and additional on-chip microcode or a great deal of hardware.

Next, secondary processor 14 turns on the trace bit in the program status register location and toggles instruction override latch 32 on. Operand override latch 34 is left off or reset. Secondary processor 14 now does a write to the processor control transfer latch 60 address, thereby passing control back to primary processor 12. Primary processor wakes up when its BGACK pin is set low and immediately completes the read command that was left in its input pipeline. When the defined address is read, along with other private storage information just left for it by secondary processor 14, primary processor 12 is able to start right up without further housekeeping. It sets its instruction fetch line in going out to the main memory where it has been incorrectly told that the move character instruction resides. However, output line 18a of OR gate 56 has now been set by the instruction override latch 32 and the instruction fetch control line 48. This means that bit 19 of the main store address put on bus 18 by primary processor 12 will be set by OR gate 58. This will cause the memory access initiated by primary processor 12 to be directed to control storage 26 without any action being taken on the part of primary processor 12 and transparent thereto.

When the modified move character instruction is fetched, since operand override latch 34 is reset and instruction fetch control line 48 is at a logical 0, the memory access made to actually move data about in accordance

with the instruction, is directed to and takes place in main memory module 24. After the move character instruction is executed, since the trace bit in the program status register as returned to primary processor 12 was set, control is again returned to secondary processor 14. It examines the return code and sees a trace bit and execute complete reason for control transfer. In response, secondary processor 14 turns off the instruction override latch 32 and restores the original program counter contents that it had initially received, when the execute instruction was encountered, to private storage 30. Control is again returned to the primary processor 12 which picks up its restored original program counter contents and all other necessary parameter and register information, and fetches the instruction that follows the execute instruction. It should be noted, depending upon the instruction to be executed by primary processor 12 when it receives control that an ensuing memory access can be to control store for both instruction fetch and operand access or for operand access only, in accordance with the output state of the override latches 32 and 34.

C L A I M S

1. Data processing system (10) for emulating an instruction set, with at least two microprocessors (12, 14), a main storage (24) and a control storage (26), all interconnected by bus means (18, 22)

characterized in that

memory access steering means (32, 34, 52, 54, 56, 58) are provided which are activated and deactivated by processor signals (48, 50) in accordance with the memory access to be performed to alter the address on the bus means and thereby to direct the memory access to the main storage or the control storage.

2. Data processing system according to claim 1, characterized in that the memory access steering means include latches (32, 34) addressable in the memory mapped private storage of at least one of the processors.

3. Data processing system according to claim 2, characterized in that the latches can be activated for instruction fetching and operand acquisition memory access.

4. Data processing system according to claim 2 or 3, characterized in that the output of the latches, is connected to gating means (58) for toggling the state of address lines in the bus means in accordance with the state of the latches.

5. Data processing system according to one of the claims 1 to 4, characterized in that the first microprocessor (12) performs a subset of the system functions, including the activation of the second microprocessor (14) and that the control store (26) is associated with the second microprocessor.

6. Data processing system according to claim 5, characterized in that the latches are addressed in the control store address space of the second microprocessor.

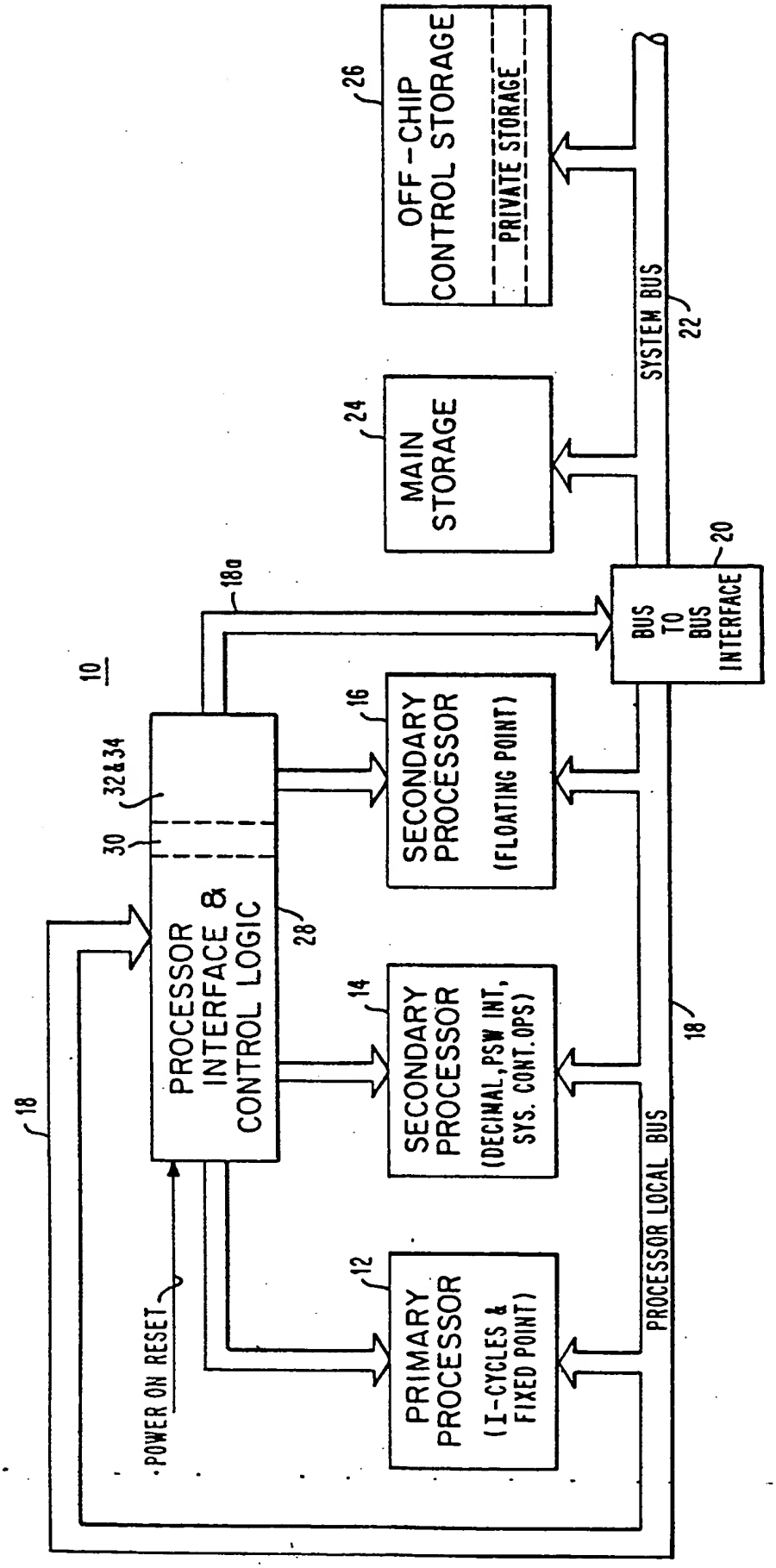
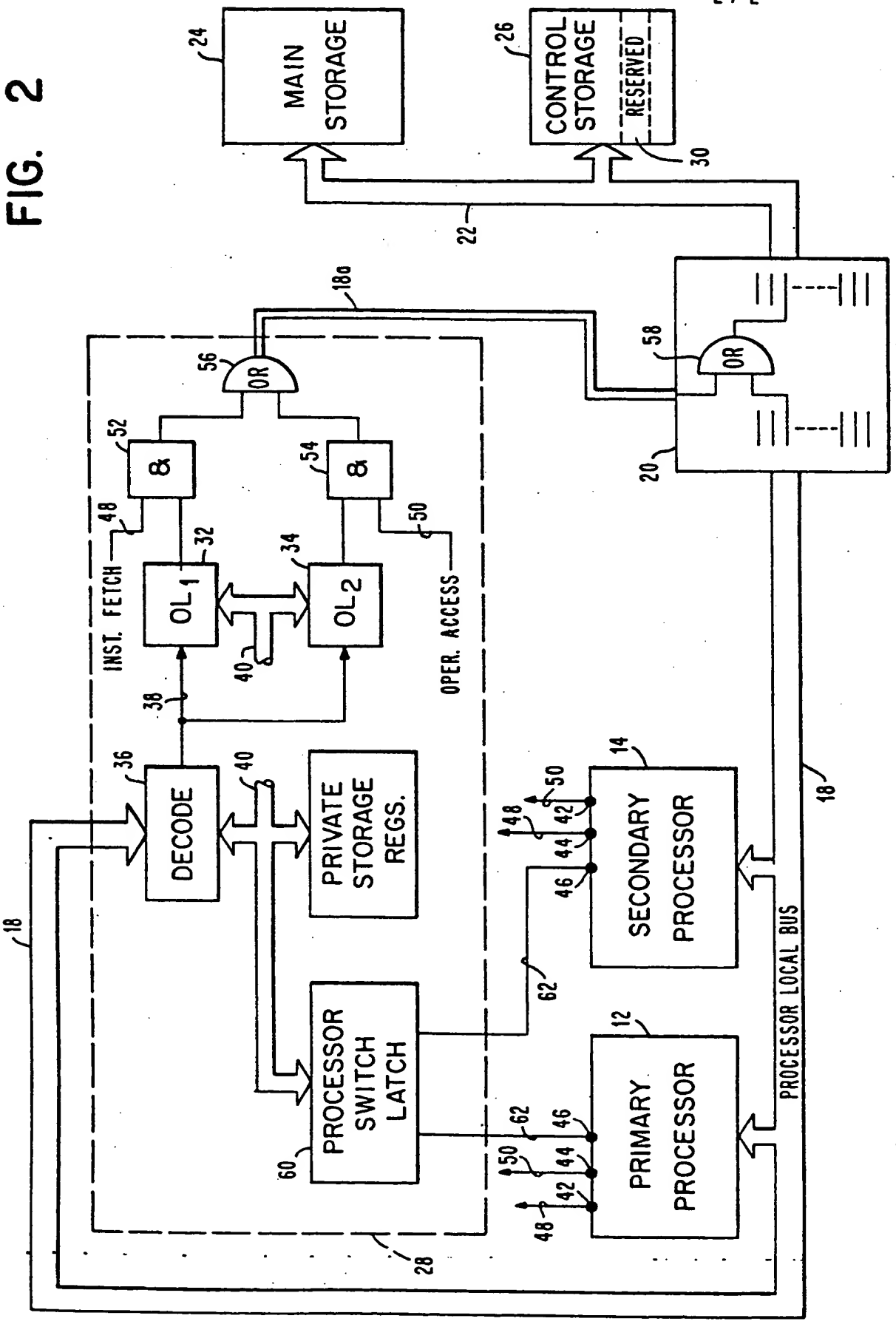

10
18
18a
18
PROCESSOR INTERFACE & CONTROL LOGIC
28
30
32&34
PRIMARY PROCESSOR (I-CYCLES & FIXED POINT)
12
SECONDARY PROCESSOR (DECIMAL, PSW INT, SYS. CONT. OPS)
14
SECONDARY PROCESSOR (FLOATING POINT)
16
POWER ON RESET
PROCESSOR LOCAL BUS
BUS TO BUS INTERFACE
20
MAIN STORAGE
24
OFF-CHIP CONTROL STORAGE
PRIVATE STORAGE
26
SYSTEM BUS
22

FIG. 1

FIG. 2


MAIN STORAGE
CONTROL STORAGE
RESERVED
OR
&
INST. FETCH
OPER. ACCESS
OL1
OL2
DECODE
PRIVATE STORAGE REGS.
PROCESSOR SWITCH LATCH
SECONDARY PROCESSOR
PRIMARY PROCESSOR
PROCESSOR LOCAL BUS